

How to Use this Template

1. Make a copy [File → Make a copy...]
 2. Rename this file: “**Capstone_Stage1**”
 3. Replace the text in green
-

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Main Screen](#)

[Edit Recipe Screen](#)

[Recipe Detail Screen](#)

[Shopping List Tab](#)

[Widget](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any edge or corner cases in the UX.](#)

[Describe any libraries you'll be using and share your reasoning for including them.](#)

[Describe how you will implement Google Play Services or other external services.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Create Main Fragments](#)

[Task 3: Create Database](#)

[Task 4: Create Activity and Fragment of the Recipe Detail Screen](#)

[Task 5: Implement navigation](#)

[Task 6: CRUD](#)

[Task 7: Adaptive Design](#)

[Task 8: Implement GooglePlayServices](#)

[Task 9: Create the Widget](#)

[Task 10: Testing](#)

GitHub Username: [servus@alexanderauer.at](#)

Shooker (Shop and Cook)

Description

Shooker is a practical little app to collect your homemade recipes and easily create a shopping list out of the necessary ingredients. You can select the recipes you want to cook and Shooker generates the shopping list for you, no more annoying search for ingredients.

Intended User

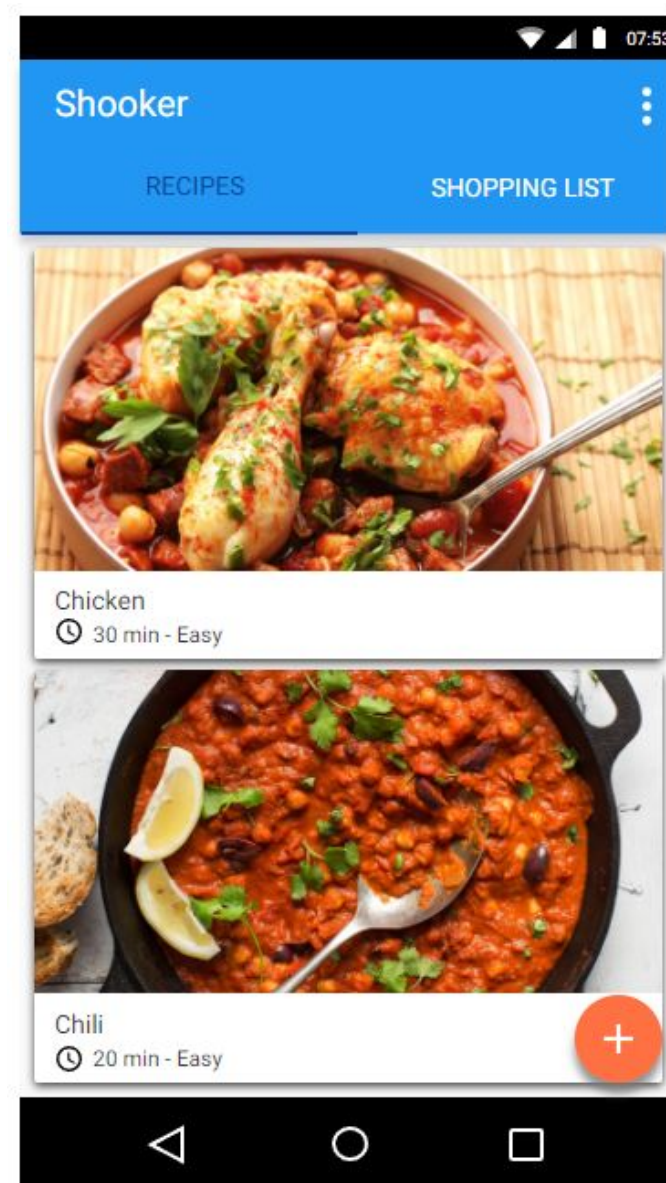
Shooker is for everyone who loves collecting and cooking recipes.

Features

- Create and delete recipes
- Create and delete ingredients/steps for recipes
- Make and assign picture to recipe (Camera)
- Mark recipe for shopping list (and vice versa)
- Generate shopping list according to ingredients of marked recipes
- Mark shopping list entries as done
- Clear entire shopping list

User Interface Mocks

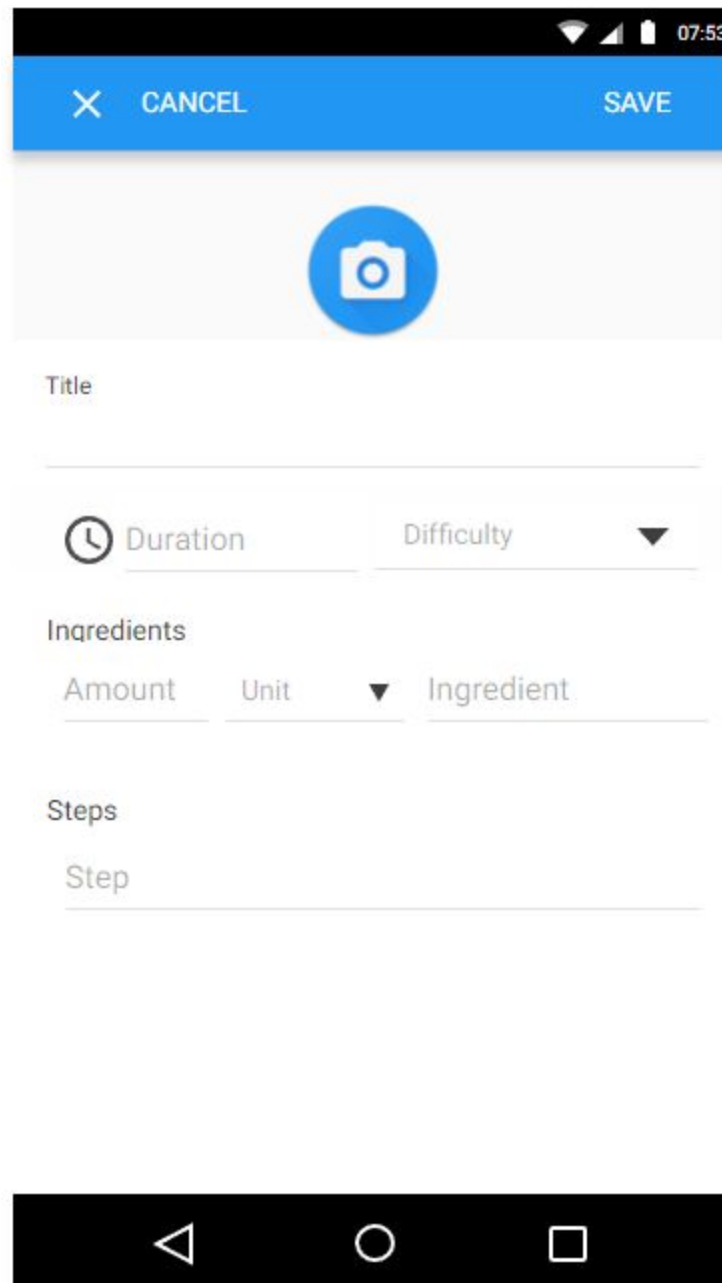
Main Screen



The Main Screen contains a TabLayout with two tabs (RECIPES and SHOPPING LIST) and a button to add new recipes. The recipe tab contains a GridView showing foto, name, duration and difficulty of all recipes. When you press the add-button a dialog appears to create a new recipe.


Users can also sort their recipes after distinct criteria.

Edit Recipe Screen



07:53

✕ CANCEL SAVE



Title

Duration Difficulty ▼

Ingredients

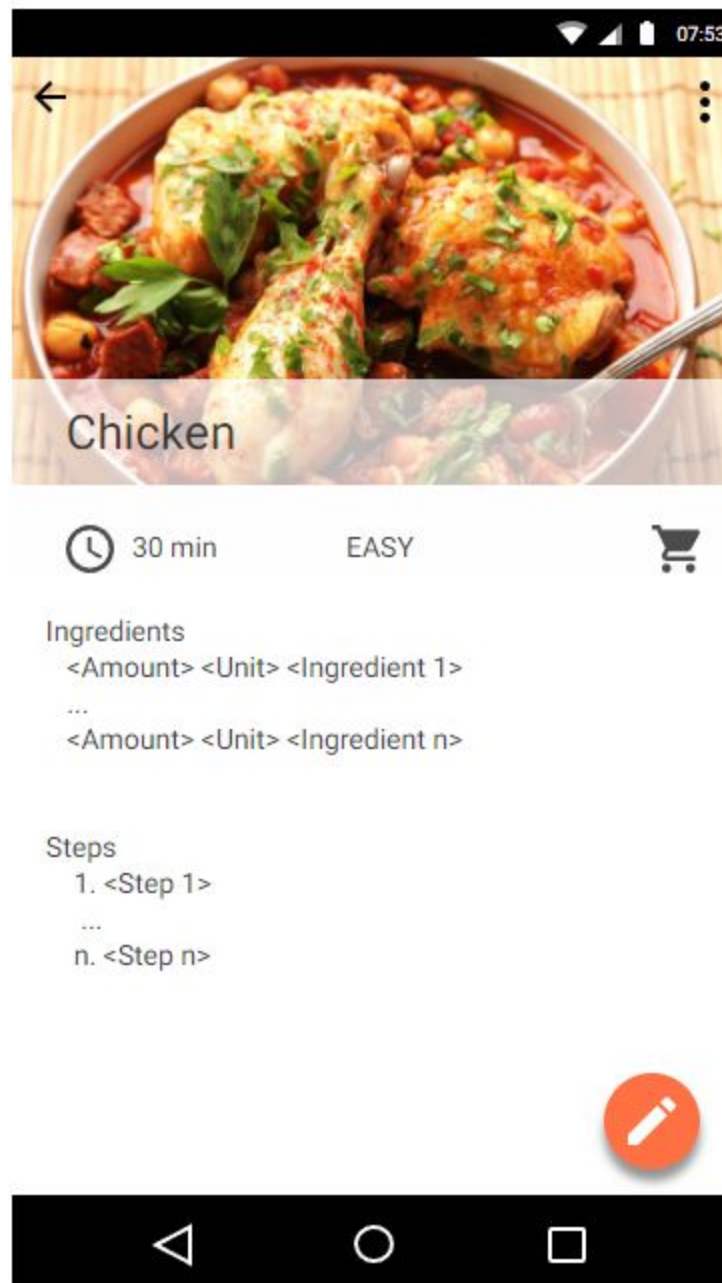
Amount	Unit	▼	Ingredient
--------	------	---	------------

Steps

Step

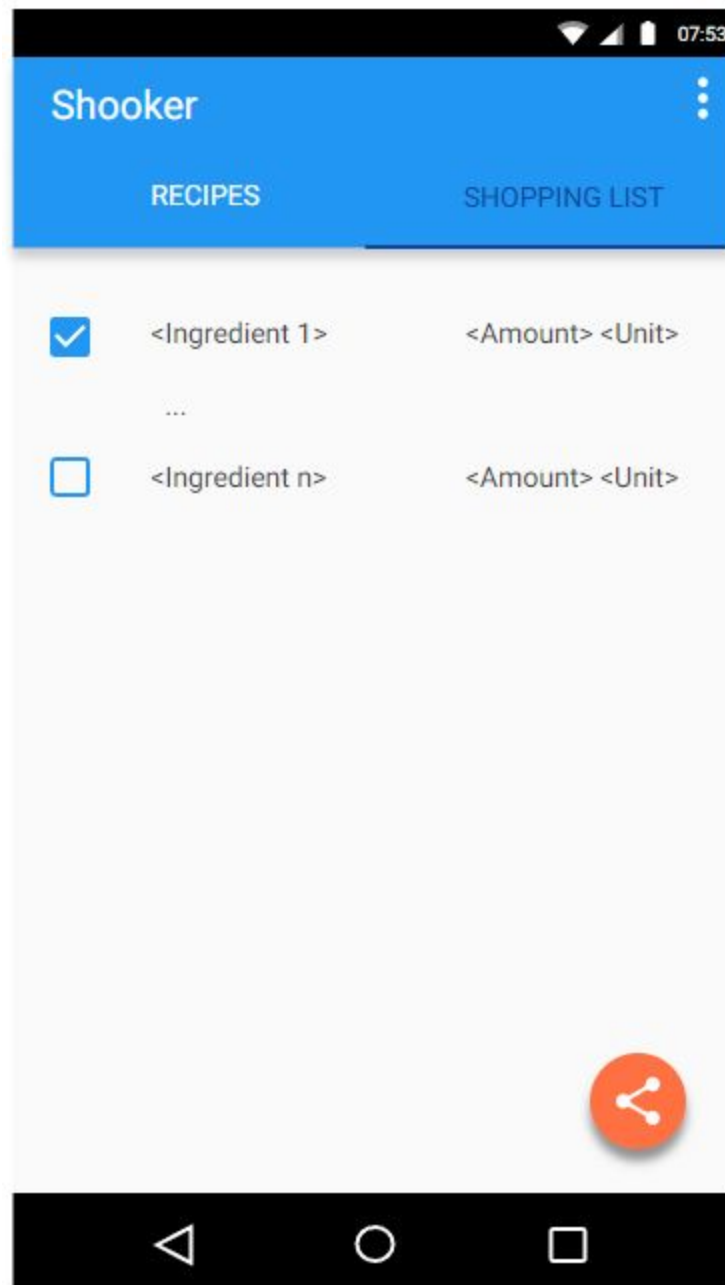
In this screen the user can enter foto, title, difficulty, duration, ingredients and steps and save the recipe. After saving, the app switches to display mode.

Recipe Detail Screen



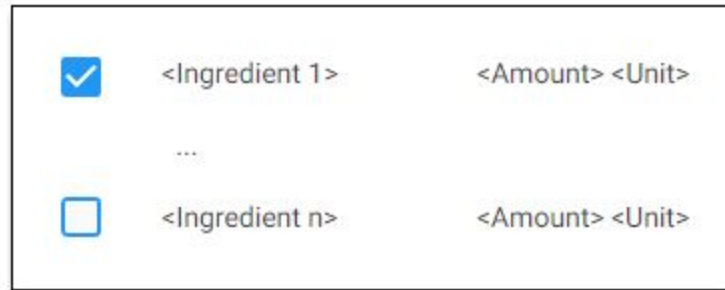
The recipe detail screen provides the recipe information in a collapsing toolbar layout. In addition it provides functions to edit, delete and add the recipe ingredients to the shopping list.

Shopping List Tab



The shopping list provides a simple view containing the ingredients of all previously selected recipes. Furthermore users can clear or share the shopping list.

Widget



The widget contains the shopping list. Users can modify the shopping list by selecting and deselecting the items.

Key Considerations

How will your app handle data persistence?

The app stores the data locally on the device in a SQLite database. Various tables are needed to store:

- Recipes
- Ingredients
- Steps

It also contains a Content Provider to manage data access.

Describe any edge or corner cases in the UX.

The Recipe Detail Activity uses two fragments, one for editing and one for displaying the recipe. When users press the Edit-FloatingActionButton, the activity replaces the fragment. By clicking the Save-Button in the edit screen, the fragment gets replaced again.

Users can mark the recipes they want to cook via a button in the recipe detail. The consequence of this action is that the ingredients of the recipe gets added to the shopping list.

Users can also empty the shopping list via the options menu. By doing so every selected recipe gets deselected which has the consequence that the shopping list gets cleared.

Describe any libraries you'll be using and share your reasoning for including them.

Libraries i want to use:

- **com.android.support:support**
- **com.android.support:appcompat**
To support backward-compatibility
- **com.android.support:design**
To be able to use controls like AppBarLayout, CollapsingToolbarLayout or CoordinatorLayout
- **com.android.support:cardview-v7**
to display the recipes in card views
- **com.android.support:recyclerview-v7**
To recycle the recipe cards
- **com.android.support:palette-v7**
To set an appropriate color for the toolbar in the recipe detail screen
- **com.android.support.constraint:constraint-layout**
To design the recipe card and the recipe detail fragment

ThirdParty: **Picasso** or **Espresso** (or both)

Describe how you will implement Google Play Services or other external services.

I want to use Admob and Analytics.

Admob is used to display commercials in the shopping list.

Analytics is used to track what people add to their shopping list.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

Since my main screen contains a tabbed layout, i will start creating an Android project from the Tabbed Activity template. Then i will include all dependencies in the build.gradle file.

Task 2: Create Main Fragments

Next i will create the two main fragments of the ViewPager. Since the RECIPE tab has to load data i need a database and a Loader.

Task 3: Create Database

My third step will be the creation of the SQLite database which will store recipes and their ingredients and steps. Thus i need three tables.

After this i will implement the ContentProvider.

Finally i can implement the Loader in the RECIPE tab.

Task 4: Create Activity and Fragment of the Recipe Detail Screen

Create and implement all necessary files.

- RecipeDetail Activity + layout xml
- RecipeDetail Fragment + layout xml
- EditRecipe Fragment + layout xml

Implement Loader for recipe details -> load ingredients and steps.

Task 5: Implement navigation

Implement navigation between activities.

Task 6: CRUD

Call CRUD methods to store data in the database. This operations will be executed in an **IntentService** so that they doesn't interrupt the main thread.

Task 7: Adaptive Design

Add some transitions and enhance Design. Make everything pretty.

Task 8: Implement GooglePlayServices

Integrate Abmob and Analytics to the shopping list. Integrate firebase and so on ...

Task 9: Create the Widget

Create a widget containing the shopping list of the selected recipes. Users can edit the shopping list via the widget (select and deselect items).

Task 10: Testing

Test the whole App

Add as many tasks as you need to complete your app.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
 - Make sure the PDF is named "**Capstone_Stage1.pdf**"

- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
- Add this document to your repo. Make sure it’s named “**Capstone_Stage1.pdf**”